Wireless Personal Communications manuscript No.
(will be inserted by the editor)

An Aggregate MapReduce Data Block Placement strategy for Wireless IoT Edge Nodes in Smart Grid

Nawab Muhammad Faseeh Qureshi · Isma Farah Siddiqui · Mukhtiar Ali Unar · Muhammad Aslam Uqaili · Choon Sung Nam · Dong Ryeol Shin · Jaehyoun Kim · Ali Kashif Bashir · Asad Abbas

Received: date / Accepted: date

N.M.F. Qureshi
Department of Computer Education, Sungkyunkwan University, Seoul, South Korea
E-mail: faseeh@skku.edu

I.F. Siddiqui
Department of Software Engineering, Mehran University of Engineering and Technology, Jamshoro, Pakistan
E-mail: isma.farah@faculty.muet.edu.pk

M.A. Unar
Department of Computer Systems Engineering, Mehran University of Engineering and Technology, Jamshoro, Pakistan
E-mail: mukhtiar.unar@faculty.muet.edu.pk

M.A. Uqaili
Department of Electrical Engineering, Mehran University of Engineering and Technology, Jamshoro, Pakistan
E-mail: vc@admin.muet.edu.pk

C.S. Nam
Department of Software, Sungkyunkwan University, Suwon, South Korea
E-mail: namgun99@gmail.com

D.R. Shin
Department of Software Engineering, Sungkyunkwan University, Suwon, South Korea
E-mail: drshin@skku.edu

J. Kim
Department of Computer Education, Sungkyunkwan University, Seoul, South Korea
E-mail: jaekim@skku.edu

A.K. Bashir
Faculty of Science and technology, University of the Faroe Islands, Faroe Islands
E-mail: alib@setur.fo

A. Abbas
Department of Computer Science and Engineering, Hanyang University ERICA, Ansan, South Korea
E-mail: asadabbas@hanyang.ac.kr

Abstract Big data analytics has simplified processing complexity of large dataset in a distributed environment. Many state-of-the-art platforms i.e. smart grid has adopted the processing structure of big data and manages a large volume of data through MapReduce paradigm at distribution ends. Thus, whenever a wireless IoT edge node bundles a sensor dataset into storage media, MapReduce agent performs analytics and generates output into the grid repository. This practice has efficiently reduced the consumption of resources in such a giant network and strengthens other components of the smart grid to perform data analytics through aggregate programming. However, it consumes an operational latency of accessing large dataset from a central repository. As we know that, smart grid processes I/O operations of multi-homing networks, therefore, it accesses large datasets for processing MapReduce jobs at wireless IoT edge nodes. As a result, aggregate MapReduce at wireless IoT edge node produces a network congestion and operational latency problem. To overcome this issue, we propose Wireless IoT Edge-enabled Block Replica Strategy (WIEBRS), that stores in-place, partition-based and multi-homing block replica to respective edge nodes. This reduces the delay latency of accessing datasets for aggregate MapReduce and increases the performance of the job in the smart grid. The simulation results show that WIEBRS effective decreases operational latency with an increment of aggregate MapReduce job performance in the smart grid.

Keywords Wireless IoT edge node · HDFS · Smart grid · Hadoop · Aggregate MapReduce block placement.

1 Introduction

Big data processing has resolved large dataset management challenges in a distributed parallel environment [1]. We find many large dataset management systems i.e. Cloudera [2], MapR [3] and Hadoop [4] in todays market that support multihoming aggregate MapReduce processing. Apache Hadoop is an open-source data management system that processes large-scale datasets in distributed environment. It consists of four main components i.e. Hadoop-common, YARN [5], HDFS [6] and MapReduce [7]. Hadoop-common is a library that provides environment functions for cluster processing. Yet Another Resource Negotiator (YARN) is the brain of Hadoop that schedules tasks and allocate resources into them. Hadoop Distributed File System (HDFS) is a file system that manages I/O operations of files and blocks in the cluster. MapReduce is an open-source programming model that processes large-scale datasets in the distributed parallel environment. HDFS comprises of three components i.e. client, Namenode, and Datanode. A client submits an input of MapReduce job and requests Namenode to allocate resources and schedules tasks over a Datanode. The Datanode processes job and generates an output into storage media of HDFS [8],[9] as shown in Fig. 1.

The smart grid is an evolution in traditional power grid architecture and adopts processing structure of big data to manage and process large volumes

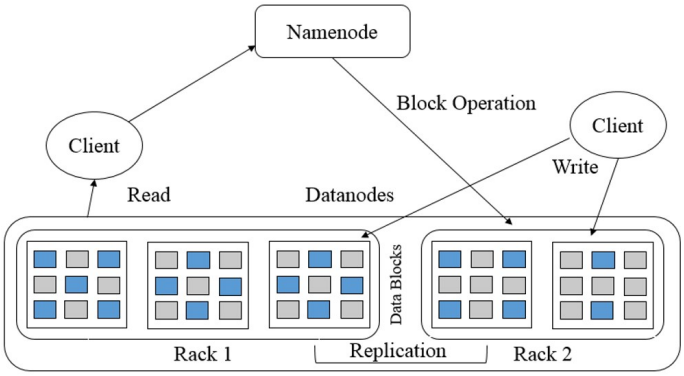

Fig. 1 HDFS Architecture

of data into distributed ends [10]. The grid supports aggregate programming and facilitates MapReduce paradigm to run aggregate functions for evaluating jobs in distributed ends [11]. This shifts consumption of resources i.e., computing capacity and memory usage from the level of the central grid to individual edge nodes and effectively performs data analytics in smart grid [12]. However, a network of the grid pays trade-off to this benefit and consumes huge bandwidth in transporting enormous size datasets for aggregate MapReduce processing [13]. Moreover, aggregate function consumes an operational latency Latency_n=Network_i (Pathdistance/(Processing Time)) in receiving data blocks through multi-homing environment [14]. Thus, aggregate MapReduce produces operational latency problems and network congestion issues in smart grid.

To resolve this issue, we propose Wireless IoT Edge-enabled Block Replica Strategy (WIEBRS), that stores data block replicas into in-place wireless IoT edge node, partition-based group of nodes and multi-homing based network of nodes and perform aggregate MapReduce job over them. This tremendously reduces network workload of moving large datasets and reduces operational latency in the smart grid.

The main contribution of the WIEBRS is:

– A novel in-place, partition-based and multi-homing based replica generation strategy.

The remaining paper is organized as follows. Section II briefly explains proposed strategy WIEBRS. Section III presents experimental environment and evaluation result. Finally, section IV describes conclusion and future research directions.

2 Wireless IoT Edge-enabled Block Replica Strategy (WIEBRS)

WIEBRS is an adaptive block replica strategy that preserves 'n+1' replica into in-place storage media and exchanges 'n+2(n)' replicas into partition 'k'

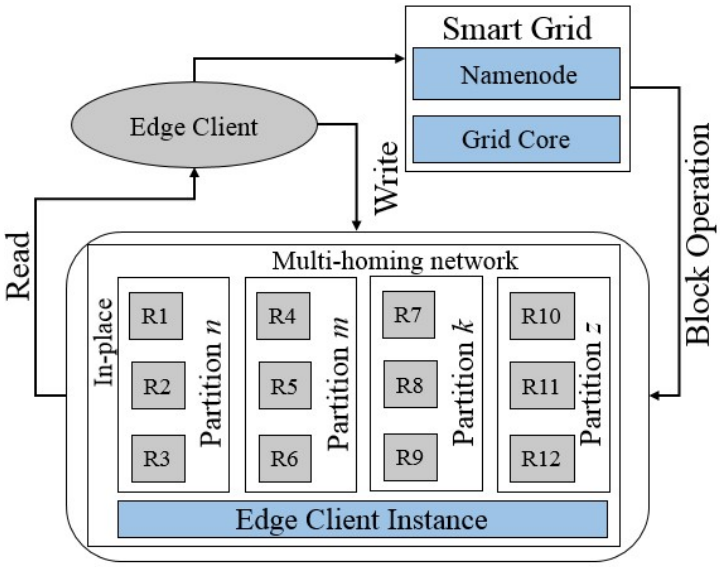

Fig. 2 WIEBRS Replica management

and 'n+3(n)' replicas to multi-homing partition in smart grid as shown in Fig. 2. WIEBRS classifies three types of replica generation strategies i.e. (i) In-place replica management, (i) Partition-based replica management and (iii) multi-homing based replica management.

2.1 In-place replica management

When an edge node processes an aggregate MapReduce job, Namenode generates in-place input split programs 'm' and performs map operation map(m). The edge node produces a map result and returns combiner task for reduce operation. Unlike the default approach, Namenode then assigns reduce operation to same edge node and produces an output into storage media. This aggregate MapReduce job processing generates an in-place output into storage media of node 'c'. The number of in-place replicas can be obtained as,

$$Replica_{in-place} = c(n+1) \tag{1}$$

2.2 Partitioned-based replica management

The partitions of wireless IoT edge nodes are designed to facilitate aggregate MapReduce job processing. Therefore, when an edge node produces an in-place replica, partition k receives a replica and exchange it with other edge nodes of the partition. The number of partitioned-based replicas can be obtained as,

$$Replica_{partition} = Replica_{in-place}(n + 2(n)) \tag{2}$$

Machine	Specifications	No. of VM	
Intel Xeon E5-2600 v2	8 CPUs, 32GB memory, 1T Disk and 128 GB SSD	3	1 Master Node, 2 Datanodes
Intel core i5	4 Core, 16GB memory, 1T Disk and 128 GB SSD	2	2 Datanodes
Hadoop	Hadoop-2.7.2 (stable)		
Virtual Machine Management	VirtualBox 5.0.16		

Fig. 3 Cluster Configuration

Table 1 HADOOP CLUSTER VIRTUAL MACHINES CONFIGURATION

Node	CPU	Memory	Disk	Configuration
Master Node	6	16 GB	HDD & SSD	Intel Xeon
Slave1	2	4GB	HDD & SSD	Intel Xeon
Slave2	2	4GB	HDD & SSD	Intel Core i5
Slave3	2	4GB	HDD & SSD	Intel Core i5
Slave4	2	4GB	HDD & SSD	Intel Core i5

2.3 Multi-homing based replica management

The term multi-homing refers to an interaction of operation between '2(w)' networks. When a partition produces a replica, it is exchanged with one or more than one multi-homing partitioned-based network. The number of multi-homing replicas can be obtained as,

$$Replica_{Multi-homing} = Replica_{in-place}(n + 3(n)) \tag{3}$$

3 Experimental Evaluation

In this section, we evaluate our proposed approach over cluster configuration as seen from Table in figure below.

3.1 Environment

The cluster configuration consists of Intel Xeon processor with 8 CPUs, 32GB memory, and storage device i.e. 1TB Hard disk drive. In addition to that, we use Intel core i5 with 4 Core, 16GB memory and storage device i.e. 1TB Hard disk drive. We install 5 virtual machines having VirtualBox 5.0.16, as seen from Table. 2.

3.2 Experimental Dataset

The experimental dataset consist of 25 data blocks of 64MB (1.56GB size).

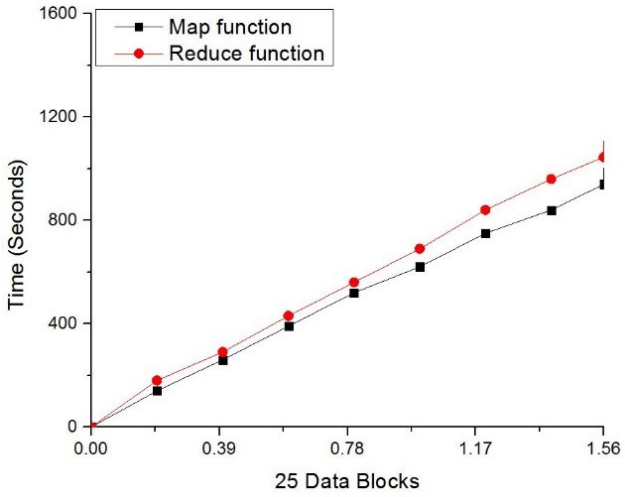

Fig. 4 Aggregate MapReduce performance in single Wireless IoT edge node

3.3 Experimental Results

The evaluation and simulations performed for evaluating proposed approach are: (i) In-place aggregate MapReduce, (ii) Partitioned-based aggregate MapReduce, and (iii) Multi-homing based aggregate MapReduce processing.

In-place aggregate MapReduce Processing MapReduce generates a single input split program due to operations being carried into single wireless IoT edge node 'c'. WIEBRS observes that single edge node consumes in-place computing capacity, memory usage and network I/O between 65 resources 75 node percentile and in-place bandwidth between 0.2 Bandwidth 0.8 GB/s for generating an output of the aggregate MapReduce job. The in-place block placement function stores 1.56 GB of the replica as shown in Fig. 4.

Partitioned-based aggregate MapReduce Processing MapReduce generates n+2(n) input split programs for processing a job into partition k. WIEBRS observes that partition k divides input split programs into k(n+2(n)) configuration and consumes computing capacity, memory usage and network I/O between 78 resource 87 partition percentile and partition network bandwidth between 0.3 Bandwidth 0.7GB/s for generating an output of the aggregate MapReduce job. The partitioned-based block placement function stores 1.56 GB of the replica to each node of partition k as shown in Fig. 5.

Multi-homing based aggregate MapReduce Processing MapReduce generates n+3(n) input split programs for processing a job into multi-homing network G. WIEBRS observes that multi-homing network divides input split programs

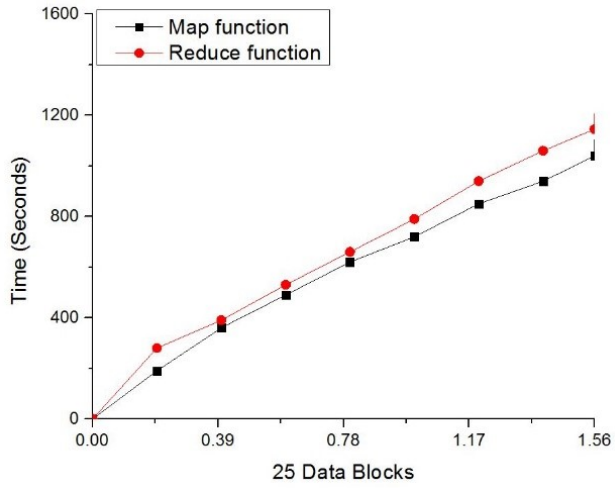

Fig. 5 Aggregate MapReduce performance in partition k

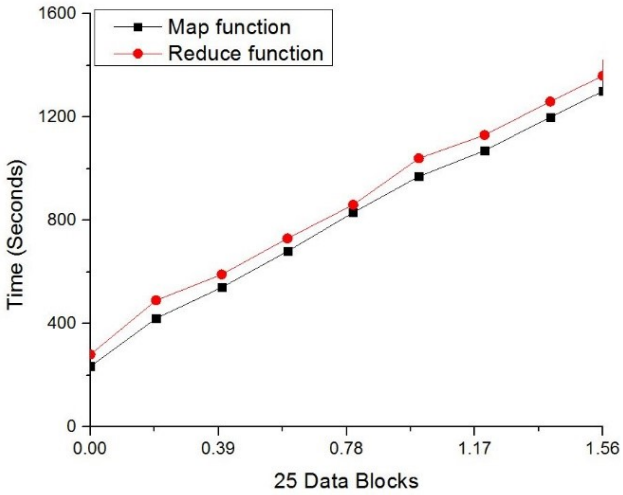

Fig. 6 Aggregate MapReduce performance in multi-homing network G

into G(n+3(n)) configuration and consumes computing capacity, memory usage and network I/O between 80 resource 88 multi-homing network percentiles and a multi-homing network bandwidth 0.6 Bandwidth 10GB/s for generating output of aggregate MapReduce job. The multi-homing network based block placement function stores 1.56 GB of the replica to each node of network G, as shown in Fig. 6.

4 Conclusion

This paper proposes Wireless IoT Edge-enabled Block Replica Strategy (WIEBRS), that stores block replicas in in-place, partition-based and multi-homing network based storage media and perform the aggregate MapReduce job in respective. WIEBRS performed experimental evaluations and observed that Wireless IoT Enabled-Edge nodes effectively increase aggregate MapReduce performance through replica management. In future, we would focus to work over inter-media replica management of Hadoop cluster.

References

1. LaValle, Steve, et al. "Big data, analytics and the path from insights to value." MIT sloan management review 52.2 (2011): 21.
2. Cloudera, "The modern platform for data management and analytics," Cloudera, 2016. [Online]. Available: http://www.cloudera.com/. Accessed: Apr. 27, 2017.
3. M. Technologies, "Featured customers", 2016. [Online]. Available: https://www.mapr.com/. Accessed: Apr. 27, 2017.
4. "Welcome to Apache Hadoop!" 2014. [Online]. Available: http://hadoop.apache.org/. Last accessed: Apr. 27, 2017.
5. "Apache Hadoop 2.7.2 Apache Hadoop YARN," 2016. [Online]. Available: https://hadoop.apache.org/docs/r2.7.2/hadoop-yarn/hadoop-yarn-site/YARN.html. Last accessed: Apr. 27, 2017.
6. "Apache Hadoop 2.7.2 HDFS users guide," 2016. [Online]. Available: https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-hdfs/HdfsUserGuide.html. Last accessed: Apr. 27, 2017.
7. "Apache Hadoop 2.7.2 MapReduce Tutorial," 2016. [Online]. Available: https://hadoop.apache.org/docs/stable/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html. Last accessed: Apr. 27, 2017.
8. Karun, A. Kala, and K. Chitharanjan. "A review on hadoopHDFS infrastructure extensions." Information & Communication Technologies (ICT), 2013 IEEE Conference on. IEEE, 2013.
9. Tsuruoka, Yukio. "Cloud computing-current status and future directions." Journal of Information Processing 24.2 (2016): 183-194.
10. Tuballa, Maria Lorena, and Michael Lochinvar Abundo. "A review of the development of Smart Grid technologies." Renewable and Sustainable Energy Reviews 59 (2016): 710-725.
11. Gungor, Vehbi C., et al. "Smart grid technologies: Communication technologies and standards." IEEE transactions on Industrial informatics 7.4 (2011): 529-539.
12. Bera, Samaresh, Sudip Misra, and Joel JPC Rodrigues. "Cloud computing applications for smart grid: A survey." IEEE Transactions on Parallel and Distributed Systems 26.5 (2015): 1477-1494.
13. Spivak, Anton, and Denis Nasonov. "Data Preloading and Data Placement for MapReduce Performance Improving." Procedia Computer Science 101 (2016): 379-387.
14. Maheshwari, Nitesh, Radheshyam Nanduri, and Vasudeva Varma. "Dynamic energy efficient data placement and cluster reconfiguration algorithm for mapreduce framework." Future Generation Computer Systems 28.1 (2012): 119-127.